523

NPS-AS-91-011

NAVAL POSTGRADUATE SCHOOL

Monterey, California

AUTOMATED DETECTION OF NAMING CONFLICTS IN SCHEMA INTEGRATION: EXPERIMENTS WITH QUIDDITIES*

Hemant K. Bhargava
and
Renae M. Beyer

February 1991

Approved for public release; distribution unlimited.

repared for: Naval Postgraduate School
Monterey, CA 93943

NAVAL POSTGRADUATE SCHOOL
Monterey, California

RADM. R. W. West, Jr.
Superintendent

Harrison Shull
Provost

The research summarized herein was accomplished with resources provided by the Naval Postgraduate School.

Reproduction of all or part of this report is authorized.

This report was prepared by:

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b DECLASSIFICATION / DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) NPS-AS-91-011	5 MONITORING ORGANIZATION REPORT NUMBER(S) Naval Postgraduate School

6a NAME OF PERFORMING ORGANIZATION Naval Postgraduate School	6b OFFICE SYMBOL *(If applicable)* AS	7a. NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c. ADDRESS *(City, State, and ZIP Code)* Monterey, CA 93943		7b. ADDRESS *(City, State, and ZIP Code)* Monterey, CA 93943
8a. NAME OF FUNDING / SPONSORING ORGANIZATION Naval Postgraduate School	8b. OFFICE SYMBOL *(If applicable)*	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER O&MN
8c. ADDRESS *(City, State, and ZIP Code)* Monterey, CA 93943		10 SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO	PROJECT NO	TASK NO	WORK UNIT ACCESSION NO

11 TITLE *(Include Security Classification)*

Automated Detection of Naming Conflicts in Schema Integration:Experiments with Quiddities

12. PERSONAL AUTHOR(S)

Hemant K. Bhargava

13a. TYPE OF REPORT Technical Report	13b TIME COVERED FROM ________ TO ______	14 DATE OF REPORT *(Year, Month, Day)* May 10, 1991	15 PAGE COUNT 30

16 SUPPLEMENTARY NOTATION

Schema Integration, Database Integration, Naming Conflicts, Unique names violations, Quiddities

17 COSATI CODES			18 SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
FIELD	GROUP	SUB-GROUP	Schema Integration, Database Integration, Naming Conflicts, Unique names violations, Quiddities

19 ABSTRACT *(Continue on reverse if necessary and identify by block number)*

This paper discusses experiments involving a mathod for the automatic detection, prior to the integration of database schemas, of conflicts in the naming of data elements within these schemas. The method relies on the representation of semantic information (called quiddity) about the data elements present in the various schemas. We develop serveral inference procedures which, utilizing this information, determine wether two distinctly named elements in fact represent the same object, or if elements with the same name actually represent different objects. The experiments are concerned with a) examining the accuracy and consistency with which quiddities of data elements might be declared by different database designers, and b) evaluating the accuracy and errors of thes automated procedures. Our results indicate that the method has promise for use in detection of naming conflicts, and that certain inference procedures are superior to others in terms of their accuracy and error rates.

20 DISTRIBUTION / AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFEIED	
22a NAME OF RESPONSIBLE INDIVIDUAL Hemant K. Bhargava	22b TELEPHONE *(Include Area Code)* (408) 646-2264	22c OFFICE SYMBOL AS/Bh

Previous editions are obsolete. SECURITY CLASSIFICATION OF THIS PAGE

S/N 0102-LF-014-6603

Automated Detection of Naming Conflicts in Schema Integration: Experiments with Quiddities*

Hemant K. Bhargava
Renae M. Beyer
Naval Postgraduate School
Monterey, CA 93943–5000

May 10, 1991

Abstract

This paper discusses experiments involving a method for the automatic detection, prior to the integration of database schemas, of conflicts in the naming of data elements within these schemas. The method relies on the representation of semantic information (called *quiddity*) about the data elements present in the various schemas. We develop several inference procedures which, utilizing this information, determine whether two distinctly named elements in fact represent the same object, or if elements with the same name actually represent different objects. The experiments are concerned with a) examining the accuracy and consistency with which quiddities of data elements might be declared by different database designers, and b) evaluating the accuracy and errors of these automated procedures. Our results indicate that the method has promise for use in detection of naming conflicts, and that certain inference procedures are superior to others in terms of their accuracy and error rates.

*We would like to express our thanks to Jim Connor, Mark Greer, Rafael Gacel, Nancy Reinhart, Jane Smith, and Barbara Treharne for making the experiments a success.

1 Introduction

Successful integration of multiple database schemas with overlapping domains requires the identification and resolution of conflicts in the naming of data elements within the schemas of these databases. This paper describes a method for the automatic detection of such naming conflicts, and presents the results of a first set of experiments involving the application of this method. The method being tested builds upon a method for detecting similar conflicts in the integration of multiple mathematical models (see [4]). It relies on the representation of certain semantic information (called *quiddity*[1]), not captured in data dictionaries, about the data elements or attributes present in the schemas being integrated. The first part of our experiments is concerned with examining the accuracy and consistency with which quiddities of data elements might be declared by different database designers. The second part of our experiments is concerned with an analysis and comparison of the accuracy and errors of a set of alternative procedures which we have developed. These inference procedures utilize the quiddity information to automatically determine whether two distinctly named elements in fact represent the same object (the *synonym* problem), or if data elements with the same name actually represent different objects (the *homonym* problem).

It is recognized in the database literature that naming problems must be detected and resolved prior to schema integration [6, 16], and that the detection of these problems is extremely tedious, time-consuming, and error-prone [2, 11, 15]. Further, several methodologies and guidelines have been proposed for the identification (and even prevention) of such conflicts [7, 12].

[1] From the *Oxford English Dictionary*, quiddity is "The real nature or essence of a thing; that which makes a thing what it is." [4]

The process is facilitated with automated tools, largely by providing quick, on-line access to information about these elements. However, a significant part of the effort, that of verifying the existence of a conflict for each pair of data elements, must still be borne by the database designer. There is also not, to our knowledge, much empirical evidence regarding the usefulness of these methodologies. Other automated tools are proposed to support the *resolution* of conflicts (e.g., see [5, 7, 10, 12]), but we will not have anything further to say about this issue, since our focus is on the *detection* of these conflicts. We will also not be concerned here with other kinds of conflicts (e.g., structural conflicts—see [11, 16, 17]) that must also be resolved prior to database integration.

More recently, the problem of naming conflicts has been addressed in the model management literature in the context of conflicts in the naming of modeling variables. The violation of the *unique names assumption*[2] in the naming of variables in multiple models causes problems when these models are integrated. Bhargava et al. [4] argued that the detection of such naming problems requires knowledge about what these variables represent, and that such knowledge must be represented formally if the detection of naming conflicts is to be automated. They proposed that modeling variables be further defined in terms of their quiddities, dimensions, and units of measurement, and illustrated with several examples how this information would be useful in detecting naming conflicts. They discussed a formal representation for quiddities, and suggested that two variables (named distinctly) be considered as candidates for a possible violation of the unique names assumption if

[2] It is often useful and convenient to assume in software systems that every individual has at most one name, unless stated otherwise. This assumption is called the *unique names assumption* [9].

they had the same quiddity and dimension. We present a summary of their approach in §2.1.

This approach raises several questions regarding the use of quiddities for the detection of unique names violations. For example, can people define quiddities correctly? Do quiddities capture sufficient information to ensure detection of these violations? What procedures must be designed to make this detection, and how accurate will they be? Early in our research, we conducted a preliminary experiment, involving a group of six subjects, in which we examined the clarity and feasibility of the concept of quiddities. We discuss this experiment and its results in §2.2. We used these results, as well as a careful analysis of the idea of quiddities as it applies to the database world, to substantially refine the concept, and to develop a set of guidelines that would assist a database designer in correctly defining quiddities of data elements (see §2.3). We then developed several alternative inference procedures that utilize quiddity information in the detection of naming conflicts. These procedures, and the rationale for each of them, are discussed in §3. Finally, we conducted an experiment to gather information about how successful this approach might be in detecting naming conflicts. Specifically, we were concerned with two questions. First, was our concept of quiddities, and our guidelines for developing them, clear and precise enough so that two different individuals would develop equivalent quiddities for the same element? Second, could this information be gainfully employed to automate the detection of naming conflicts, and if so, what would be the accuracy and error rates of the various inference procedures? We examine these questions and our experiment in §4.

2 Capturing Semantic Information about Data Elements using Quiddities

In this section, we explain the concept of quiddities, as proposed by Bhargava et al., and as refined by us, and present certain guidelines that we believe will facilitate the correct declaration of quiddities.

2.1 Model Integration: Unique Names Violations and Quiddities

The violation of the unique names (of modeling variables) assumption causes a problem in model integration similar to the one caused by naming conflicts in database integration. The homonym and synonym problems are both special cases of unique names violations (UNVs). After an analysis of information requirements for detecting UNVs, Bhargava et al. concluded that such detection required descriptive information about the variables, and that this information be represented using a descriptive apparatus that was sufficiently rich and unambiguous. They suggested that the *quiddity* and *dimension* of variables be represented formally, and argued that if two distinctly named variables had equivalent dimensions and quiddities, then those variables possibly constituted a UNV.

The quiddity of a modeling variable (or data element) provides a description of "what it is the variable is about", and in particular, information that is relevant to UNV detection [4]. Bhargava et al. proposed a formal language for representing the quiddity of a variable. In this language, quiddity is defined in terms of six categories of information about the variable: *stuff, types of stuff, attributes of stuff, types of attributes of stuff*, and *metafunctions.* They showed, using several examples, that these six categories were required to be able to distinguish the quiddities of variables that *appeared* the same but were really not (and so did not pose a UNV problem). The quiddity expression for a variable is developed by specifying terms, from a given vocabulary, for (some, or all, of) these categories, and combining these terms according to the grammar for the language. These components, and the quiddity, are illustrated using the following two examples. (For our purpose, the rules of formation for representing quiddity expressions in the formal language are not relevant, and will not be disucssed here.)

Example 1 *tail-number*

Description*:* Tail number of a fighter aircraft.

Quiddity*: tail-number(fighter(aircraft))*

Example 2 *command-com*

Description*:* Indicates whether or not damage is caused by a virus to an operating system.

Quiddity*: indicator(damage(virus,system))*

The component *stuff* answers the question "What is the object this variable is about?" Stuff is usually, but not necessarily, indicated by a noun, describing individual things or collections of individual things, such as cars, trucks, or ships. In examples 1 and 2 above, the stuff terms are *aircraft* and *damage*, respectively. A stuff term may have an associated *arity* if one or more *arguments* are required to fully define it. In example 2 above, we are interested in damage *by* something (virus) *to* something (system). Therefore, damage has arity 2, with the arguments *virus* and *system*. The component *stuff type* answers the question "What sort of or kind of stuff is it?" Stuff types further describe stuff. In example 1, the stuff type term *fighter* qualifies the stuff expression *aircraft*.

The component *stuff attribute* answers the question "What is it about the stuff that you are interested in?" In the above examples, we are interested in the *tail-number* of the aircraft, and whether or not (*indicator*) there is damage by the virus to the system. The component *stuff attribute type* answers the question "What sort of or kind of stuff attribute is it?" Stuff attribute types further qualify stuff attribute. None of examples 1 and 2 have a stuff attribute type, but, for instance, an attribute *cost* might be qualified as a *purchase* cost or a *production* cost.

Metafunctions capture information, usually statistical or mathematical, about the attribute of the data element. Examples of metafunctions are average, maximum, minimum, sum, and variance. None of examples 1 and 2 have a metafunction, but, for instance, an attribute *cost* might be an *average* cost or a *minimum* cost.

Bhargava et al. recognized that the quiddity expressions in their language only approximated the actual quiddities of variables. They argued that, in spite of this appoximation, the concept was sufficiently rich and expressive to be of use in UNV detection. Assuming that is true, the use of quiddities for UNV detection raises several questions. Is it possible that different people will specify a different quiddity for the same variable (even given the same information about it)? Are the quiddity categories general enough to capture relevant information in typical database applications? Are the various quiddity categories clear and meaningful? If not, which of these are not clearly understood? While our interest went beyond these issues, we conducted a small experiment to gain an understanding of the answers to these questions.

2.2 Quiddities? A Preliminary Experiment

We conducted a preliminary experiment to examine the above-mentioned issues in quiddity acquisition. Two databases, overlapping in their real world domains, were used as the basis for this experiment. These databases were developed by separate teams. Twelve data elements from each database were selected for quiddity formulation. We ensured that unique name violations did exist among the selected subsets of these two databases. Each subject was given a packet which contained the following: an overall information sheet, a basic instruction sheet, a work sheet (for practice and instructional purposes prior to beginning the experiment), a general (i.e., the terms were not separated by quiddity category) vocabulary list, a list of standard data dictionary entries for the selected data elements, and sample output reports from the databases displaying data values for the selected elements. All subjects were provided with instruction on the concept, representation, and rules of formation of, quiddities. Sample quiddity problems were discussed with the subjects prior to beginning the experiment. (See [3] for details.)

Six subjects, three for each database, participated in the experiment. Each subject was asked to independently formulate quiddities for the elements in the database assigned to the subject. Thus for each of the two databases, we had a group of three subjects formulating quiddities for the same data elements using the same set of information about these elements. The subjects were advised, though not restricted, to use the vocabulary provided in the vocabulary list.

We performed the following across-subject analyses within each group, using the quiddities formulated by the subjects. First, within each group, we compared the quiddities developed by the subjects with the "correct" quiddity (determined prior to the experiment). We found that very few quiddities were correctly defined—there were no matches for group 1 and 7 matches for group 2, out of a maximum of 36 possible matches in each group. (Two quiddity expressions matched only when they agreed on every quiddity component.) Second, for each pair of subjects in the same group, we compared the quiddities developed by those subjects. Again, we found that very few quiddities were identically defined—there were only 2 matches for group 1 and 3 matches for group 2, out of a maximum of 36 possible matches in each group.

Comparisons of individual quiddity components showed that the subjects were often not able to correctly identify the *stuff* and *stuff attribute* (the performance on the other categories was even poorer). Compared with the correct quiddities, there were 7 stuff matches in group 1, 24 stuff matches in group 2, 24 attribute matches in group 1, and 14 attribute matches in group 2, all out of a maximum of 36. Compared within the groups, the numbers were 13, 22, 16, and 13, respectively, again out of a maximum of 36. There were several cases where the subjects interchanged the stuff with the stuff attribute.

What do we learn from these results? We find, a) even though this was a small experiment, b) the subjects had only a quick introduction to the concept of quiddities, and c) we defined a "match" very strictly, that there was much confusion in applying the definition and concept of quiddity and its components. The definitions and meanings of the various categories were not sufficient or unambiguous enough for the subjects to develop quiddities in a manner consistent with the actual concepts. There was a lack of clear distinction between the stuff and stuff attribute components, the two most significant quiddity categories. This led to confusion in determining the arity of the stuff component and in identifying the sortal information provided by the stuff type and the stuff attribute type. Further, the subjects were unclear about the level of detail at which they needed to define the quiddities.

2.3 Guidelines for Developing Quiddities

We used the results of our preliminary experiment and feedback from the subjects to re-analyze the concept of quiddities. We found that it was still useful to represent quiddity in terms of the six categories discussed earlier. However, we refined our interpretations of some of these categories. We concluded that each data element must have exactly one stuff term and exactly one stuff attribute term. The stuff attribute is a measurable aspect (the thing being measured) of the stuff, and is best indicated by examining some of the data values corresponding to the data element. The stuff is then simply the thing that this measurement is about. It is particularly useful to include an attribute called *indicator*—this is useful for data elements with Boolean or similar values, which indicate the status of some property (the stuff) of the data element. We have also suggested several changes in the quiddity acquisition process based on our analysis of the information being captured in the quiddity components. For lack of space (see [3] for details), we will only summarize the results and guidelines for quiddity formulation that were derived from this analysis. These guidelines are listed below.

1. Gather Information: Examine the definition of the data element using information, such as that in the data dictionary, about the data element.

2. Examine Data: Examine a collection of actual data values, and their units of measurement (if any), for the data element. Answer the ques-

tions "What does this data represent?" "What are these values a measure of?" For example, "John" and "Mary" are *names*, $21 and $40 represent *costs*, and $\{0, 1\}$ values are *indicators* of something.

3. Identify Attribute: Identify the stuff attribute by examining the data values and data definition. The attribute is usually a noun, and is a measurable (in the abstract sense) item.

4. Identify Stuff: Now identify the stuff term by looking at the attribute term and asking the question "This is an attribute of *What*?" The stuff is also generally a noun and is the object of a prepositional phrase associated with the stuff attribute. For example, if the attribute is *cost*, the question "Cost of what?" will lead to the stuff term.

5. Identify Remaining Components. Answer the questions "What sort of stuff is it?" (the stuff type), "What sort of stuff attribute is it?" (the stuff attribute type), and "Is the stuff term a function of something else?" (stuff arguments).

6. Verify Terms: Ensure that the terms are present in the appropriate category in the vocabulary, and have the same interpretation as intended. If not, select a suitable term from the vocabulary.

3 Procedures for Determining Quiddity Equivalence

In this section, we present automated procedures for the detection of possible naming conflicts in schemas of different databases. It is useful, for this purpose, to view the *stuff*, *stuff-arguments*, and *stuff-type* terms collectively as the *stuff-part*, and to view the *stuff-attribute* and *stuff-attribute-type* terms as the *attribute-part.* Then, we define two quiddities to be equivalent if and only if they have equivalent *stuff-part*s and equivalent *attribute-part*s. We use the symbol $\equiv$ for equivalence.

We need to operationalize this definition of quiddity equivalence by defining rules for *stuff-part* and *attribute-part* equivalence. We also need rules for establishing whether or not two *terms* are equivalent. The alternative quiddity equivalence procedures we propose here, and in particular, our rules for term equivalence, stuff-part equivalence, and attribute-part equivalence, are motivated by certain of our hypotheses regarding how different people may interpret and specify quiddities. We begin by stating these hypotheses, and follow that by specifying the alternative rules and procedures.

1. *Stuff* and *Stuff-attribute* are the most significant quiddity components.

2. Different people are likely to choose terms of different specificity in defining the same quiddity. For example, one person might use the term *vehicle* for the same component for which another person chose the more specific term *truck*.

3. There is scope for confusion between the *stuff-type* and the *stuff-arguments* components.

4. Some people are likely to define quiddities more extensively than others. For example, one might use the stuff type terms *fighter* and *unmanned* to qualify the stuff term *aircraft* of example 1.

3.1 Term Equivalence

When is one term equivalent to another? Clearly, they are equivalent when they are exactly the same. They could also be considered equivalent if one is the synonym of the other. Finally, based on hypothesis 2, they could be considered equivalent if they are in the same "class," and one term is more (or, less) specific than the other. To operationalize the last two cases, we will assume that there are two relationships defined between terms in the vocabulary. First, the binary relation *synonyms*, such that *synonyms*(α, β) is true when α and β are synonyms. This relation is transitive as well as commutative. Second, the binary relation *is-a*, such that *is-a*(α, β) is true when α is a specialization of β. This relation is transitive. We write $\alpha \equiv_{T_i} \beta$ to mean that α is equivalent to β using term equivalence rule T_i. Then we define the following three alternate rules for term equivalence.

1. $\alpha \equiv_{T_1} \beta$ if α and β are syntactically the same.

2. $\alpha \equiv_{T_2} \beta$ if $\alpha \equiv_{T_1} \beta$ or *synonyms*(α, β).

3. $\alpha \equiv_{T_3} \beta$ if $\alpha \equiv_{T_2} \beta$ or *is-a*(α, β), or *is-a*(β, α).

We note that a *set* of terms is equivalent to another *set* of terms if there is some permutation of the terms in one set, such that the i^{th} term of that set is equivalent to the i^{th} term of the other (i ranging from 1 to the number of terms in the set).

3.2 Stuff-Part Equivalence

When is the stuff-part of one quiddity equivalent to the stuff-part of another? In the simplest and strictest case, when the stuff term, argument terms, and stuff-type terms, in one are equivalent to the stuff term, argument terms, and stuff-type terms, respectively, in the other. (Note that any of the three rules for term equivalence could be used in this rule, and in the remaining stuff-part equivalence rules.) Second, (motivated by hypothesis 4), when the argument terms of one are a subset[3] of the argument terms of the other, the stuff-type terms of one are a subset of the stuff-type terms of the other, and the stuff terms are equivalent. Third, (motivated by hypotheses 3 and 4), when the argument and stuff-type terms (collectively) of one are a subset of the argument and stuff-type terms of the other, and the stuff terms are equivalent. Note that, motivated by hypothesis 1, the stuff terms must be equivalent in each of these rules. Fourth, and least strictly, when the entire stuff-part of one is a subset of the stuff-part of the other.

We write $\alpha \equiv_{S^j_i} \beta$ to mean that the stuff-part α is equivalent to stuff-part β using stuff-part equivalence rule S_j in conjunction with term equivalence rule i. Then we define the following four alternate rules for stuff-part equivalence.

1. $\alpha \equiv_{S^1_i} \beta$ if

   (a) stuff(α) $\equiv_{T_i}$ stuff(β),

   (b) arguments(α) $\equiv_{T_i}$ arguments(β), and

   (c) stuff-type(α) $\equiv_{T_i}$ stuff-type(β).

2. $\alpha \equiv_{S^2_i} \beta$ if

   (a) stuff(α) $\equiv_{T_i}$ stuff(β),

[3] Since the direction of the subset relationship between two sets of terms is irrelevant in our rules, we will use the symbol $\simeq$ to mean that one is a subset of the other.

(b) arguments(α) $\simeq_{T_i}$ arguments(β), and

(c) stuff-type(α) $\simeq_{T_i}$ stuff-type(β).

3. $\alpha \equiv_{S_i^3} \beta$ if

   (a) stuff(α) $\equiv_{T_i}$ stuff(β), and

   (b) $\langle$arguments,stuff-type$\rangle(\alpha) \simeq_{T_i}$ $\langle$arguments,stuff-type$\rangle(\beta)$.

4. $\alpha \equiv_{S_i^4} \beta$ if

   (a) $\langle$stuff,arguments,stuff-type$\rangle(\alpha) \simeq_{T_i}$ $\langle$stuff,arguments,stuff-type$\rangle(\beta)$.

3.3 Attribute-Part Equivalence

When is the attribute-part of one quiddity equivalent to the attribute-part of another? In the simplest and strictest case, when the attribute term and attribute-type terms in one are equivalent to the attribute term and attribute-type terms, respectively, in the other. (Again, any of the three rules for term equivalence could be used in this rule, and in the remaining attribute-part equivalence rules.) Second, (motivated by hypothesis 4), when the attribute-type terms of one are a subset of the attribute-type terms of the other, and the attribute terms are equivalent. Note that, motivated by hypothesis 1, the attribute terms must be equivalent in both of these rules. Third, and least strictly, when the entire attribute-part of one is a subset of the attribute-part of the other.

We write $\alpha \equiv_{A_i^k} \beta$ to mean that the attribute-part α is equivalent to attribute-part β using attribute-part equivalence rule A_k in conjunction with term equivalence rule i. Then we define the following three alternate rules for attribute-part equivalence.

1. $\alpha \equiv_{A_i^1} \beta$ if

   (a) attribute(α) $\equiv_{T_i}$ attribute(β), and

   (b) attribute-type(α) $\equiv_{T_i}$ attribute-type(β).

2. $\alpha \equiv_{A_i^2} \beta$ if

   (a) attribute(α) $\equiv_{T_i}$ attribute(β), and

   (b) attribute-type(α) $\simeq_{T_i}$ attribute-type(β).

3. $\alpha \equiv_{A_i^3} \beta$ if

   (a) $\langle$attribute,attribute-type$\rangle(\alpha) \simeq_{T_i}$ $\langle$attribute,attribute-type$\rangle(\beta)$.

3.4 Inference Procedures

An inference procedure P_{ijk} is simply a combination of rules T_i, S_j, and A_k for determining term, stuff-part, and attribute-part equivalence, respectively. We write $\phi \equiv_{ijk} \psi$ to mean that the quiddity ϕ is equivalent to quiddity ψ using procedure P_{ijk}. Based on the equivalence rules discussed above, there are 36 ($3 \times 4 \times 3$) possible procedures. However, given the motivations behind the equivalence rules, only 12 procedures—P_{i11}, P_{i22}, P_{i32}, and P_{i43} ($i = 1,2,3$)—are meaningful. These equivalence rules and procedures were implemented in the Edinburgh syntax of Prolog [14] and tested on a Macintosh implementation of Prolog.

Each inference procedure will determine whether or not a pair of variables constitutes a naming conflict (homonym or synonym problem). There are two kinds of errors, called *Type 1* and *Type 2* errors, that a procedure can commit. A type 1 error occurs when the procedure indicates a UNV problem when in fact there is none. A type 2 error occurs when the procedure fails to indicate a problem when in fact there is one. The second one is the more important to avoid, since our objective is to detect UNVs. In general, let w_1 and w_2 denote the weights assigned to these two kinds of errors. (A higher weight indicates that it is more costly to commit an error, and w_2 will usually be much greater than w_1.) Suppose that for a given pair of databases, a procedure commits n_1 errors of type 1 and n_2 errors of type 2. Then the *weighted error rate* for that procedure is given by

$$E_w = n_1 w_1 + n_2 w_2 \tag{1}$$

The ratio $w = \frac{w_2}{w_1}$ is a measure of the relative weight of these two errors. It will be convenient to set w_1 to 1 (so that $w = w_2$), and to vary w_2 depending on the relative importance of avoiding type 2 errors. A procedure dominates another procedure if it commits fewer errors of both types. However, in general, if a procedure commits fewer errors of one type, it is likely to commit more errors of the other type. In that case, the weighted error rate, with a suitable choice of w_2, can be used to compare various procedures.

4 Quiddity Acquisition and Inference: An Experiment

In this section, we describe an experimental investigation of the usefulness of quiddities in the detection of naming conflicts. We first describe the experiment and its design, and then examine the results of this experiment in terms of a) the correctness of specification of quiddities, and b) the accuracy of the alternative inference procedures in the detection of naming conflicts.

4.1 Experiment Design

We conducted a second experiment to investigate the usefulness of a) our guidelines for developing quiddities (§2.3), and b) our procedures for determining quiddity equivalence (§3). The experiment involved two new databases, also developed by different teams. This experiment was designed and conducted in a manner similar to that of the preliminary experiment (§2.2), except for the following variations. Each of the databases had 15 data elements. There were 5 synonym and 3 homonym problems in these schemas. In this experiment, the vocabulary provided to the subjects was classified by quiddity category, and the subjects were restricted to using only the terms in the vocabulary.

4.2 Experiment Results: Quiddity Acquisition

The experiment results were again divided into two groups, one for each database. There are a total of 45 quiddities developed by subjects in each group, three for each of the fifteen data elements.

We performed the same across-subject analyses within each group as in the preliminary experiment. Comparing these quiddities with the correct ones, we found that few quiddities were correctly defined—there were 13 matches for group 1 and 15 matches for group 2, out of a maximum of 45 possible matches in each group. (These numbers do increase if we allow for the use of synonym terms or for the use of more or less specific terms.) Comparing quiddities for each pair of subjects in the same group, we found that few quiddities were identically defined—there were only 10 matches for group 1 and 7 matches for group 2, out of a maximum of 45 possible matches in each group. These numbers represent a significant increase over those obtained in the previous experiment.

Comparisons of individual quiddity components showed that the subjects were now usually able to correctly identify the *stuff* but were still not performing well on the *stuff attribute* (the performance on the other categories was again poorer than on these two). Compared with the correct quiddities, there were 39 stuff matches in group 1, 35 stuff matches in group 2, 25 attribute matches in group 1, and 27 attribute matches in group 2, all out of a maximum of 45. Compared within the groups, the numbers were 13, 22, 16, and 13, respectively, again out of a maximum of 45. These numbers again represent a significant improvement over the results of the previous experiment. There were very few instances in which subjects interchanged terms between stuff and attribute in this experiment.

An examination of the quiddities developed by various subjects showed that in spite of the improvements over the previous experiment, there were still inconsistencies across subjects in the specification of the stuff type, stuff arguments, and attribute type terms. These inconsistencies reflect differences in the specificity of terms chosen for quiddity components (see hypothesis 2). They also reflect uncertainty about the level of detail required in specifying a quiddity (see hypothesis 4). Some subjects demonstrated a tendency to be consistently more descriptive than others, i.e., they listed more terms for the stuff type and stuff attribute type component than other subjects.

What do these results say about the usefulness of quiddities in UNV detection? The percentages of correctly specified quiddities (and quiddity components) are still fairly low. However, these results were based on the definition of a quiddity "match" as a strict equivalence of all components, i.e., procedure P_{111} was implicitly utilized to determine quiddity equivalence. Are other procedures more appropriate for determining quiddity equivalence? Can the inconsistencies in quiddity specification be compensated for by more sophisticated quiddity equivalence procedures? We now move on to an examination of these questions.

4.3 Experiment Results: Inference Procedures

Recall that there were 15 data elements in each database schema and 3 subjects in each of the two groups. There were 5 synonym problems and 3 homonym problems in the two database schemas. We used each of the 12 inference procedures to compare quiddities developed by each of the 9 pairs $\langle s_1, s_2 \rangle$ of subjects with subject S_j belonging to group j.[4] For each comparison of pairs of data elements, each procedure determined whether or not the element names had any naming conflict. Similarly, we used each procedure to examine naming conflicts using the correct quiddities for the elements in each database.

4.3.1 Results of Procedures: Examples

We begin by illustrating the results of selected procedures on a small set of data elements. Databases 1 and 2 both contained information about courses offered at the Naval Postgraduate School (NPS). From database 1, consider elements DPT (designates a department at NPS), PREQ-DPT (code identifying a prerequisite department), and EMPH-AREA (name of an emphasis area available to students as an area of study within a particular curriculum). From database 2, consider the elements DEPT and PREREQ-DEPT (code

[4] That results in $(15 \times 15) \times 12 \times 9 = 24{,}300$ comparisons for across-subject analyses.

Data Element	Stuff	Arguments	Stuff Type	Attribute	Attribute Type
DPT	department		NPS	designator	
PREQ-DPT	department		prerequisite	identifier	
EMPH-AREA	emphasis-area		curriculum	title	
DEPT	department		NPS	identifier	
PREREQ-DEPT	department		NPS, prerequisite	identifier	
EMPH	emphasis-area		NPS	identifier	
EMPH-NAME	emphasis-area		NPS	title	

Table 1: Examples of Quiddities of selected Data Elements

	Database-1 Element	Database-2 Element	Are they Synonyms?	Detected as Synonyms by		
				P_{111}	P_{232}	P_{343}
1	DPT	DEPT	Yes	No	Yes	Yes
2	PREQ-DPT	PREREQ-DEPT	Yes	No	Yes	Yes
3	EMPH-AREA	EMPH-NAME	Yes	No	No	Yes
4	PREQ-DPT	DEPT	No	No	No	No
5	EMPH-AREA	EMPH	No	No	No	Yes
6	PROF-PHONE	SSN	No	No	No	Yes

Table 2: Examples of Synonym Detection using selected Procedures

identifying a prerequisite department at NPS), which are really synonyms for DPT and CRS respectively. From database 2, also consider elements EMPH (code identifying the emphasis area), and EMPH-NAME (title of an emphasis area that students may select at NPS). The quiddities for these elements are indicated in table 1, and the results (for synonym detection) of applying procedures P_{111}, P_{232}, and P_{343} are shown in table 2.

It would be useful for the reader to examine the data element definitions (given above) and sample quiddities (table 1) as well as the results of applying

Using Correct Quiddities

Procedure	# Synonyms Found (Hits)	Type 1 Errors	Type 2 Errors
111	1	0	4
122	5	2	2
132	5	2	2
143	5	2	2
211	3	1	3
222	11	7	1
232	11	7	1
243	11	7	1
311	13	11	3
322	31	27	1
332	31	27	1
343	44	39	0

Using Subjects' Quiddities

Procedure	# Synonyms Found (Hits)	Type 1 Errors	Type 2 Errors
111	1.0	0.0	4.0
122	2.3	1.0	3.7
132	2.3	1.0	3.7
143	2.7	1.3	3.7
211	2.9	0.7	2.8
222	11.1	7.4	1.3
232	10.8	7.1	1.3
243	13.0	9.3	1.3
311	10.4	8.2	2.8
322	35.4	31.3	0.9
332	31.9	27.8	0.9
343	46.1	41.4	0.3

Table 3: Detecting Synonym Problems (Total synonym pairs = 5)

the three procedures to the six pairs of data elements (table 2).

4.3.2 Synonym Detection

The results (for detecting synonym problems) of applying these procedures to the correct quiddities, and to quiddities developed in the experiment (the numbers represent an average over 9 comparisons) are shown in table 3.

We draw several interesting observations from these results. First, we note that as we vary the i of P_{ijk} from 1 to 2 there is an increase in the number of "hits", a decrease in type 2 errors (only 1—or 1.3—for procedures P_{222}, P_{232}, and P_{243}), and not much of an increase in type 1 errors. This happens since these procedures allow subjects to use alternate equivalent terms (i.e., synonyms—e.g., *price* and *cost*) for each quiddity component. As we vary i from 2 to 3, however, there is a huge increase in the number of hits and in type 1 errors. (There is not much scope for reduction of type 2 errors, though.) While not desirable, this is consistent with hypothesis 2. Second, as we vary j from 1 to 2 or 3, there is a decrease in type 2 errors, consistent with hypotheses 3 and 4. Similarly, as we vary k from 1 to 2, there is a decrease in type 2 errors, again consistent with hypothesis 4. Third, setting j to 4 (mixing *stuff* with *stuff type* and *stuff arguments*) and k to 3 (mixing *attribute* with *attribute type*) is not too useful since it leads to a huge increase in type 1 errors. This is consistent with hypothesis 1 which asserts that *stuff* and *attribute* are the most significant components.

In terms of the relative weight w $(= \frac{w_2}{w_1})$ we found that procedures P_{222} and P_{232} were the best procedures[5] (had the lowest E_w) for a wide range $1 \leq w \leq 25$ of w values. Only for $w > 25$ ($w > 32$ in the case of the correct quiddities), does procedure P_{343} become more attractive. (Note that values of w less than 1 are not meaningful.) This is a significant range, and suggests that P_{222} and P_{232} might be the best procedures to use for UNV detection. These procedures failed to detect one synonym problem (EMPH-AREA,EMPH-NAME), but that, we found, was an unusual case. It turned out that even though these elements referred to the same concept, their definitions in the data dictionary were quite different (see §4.3.1), and led us to include the stuff type term *curriculum* in one case, and *NPS* in another. These two procedures did succeed in pruning the detection problem from 225 pairs of data elements (15 × 15) to 11 pairs (number of hits). These results indicate the usefulness of quiddities in detection of synonym problems.

[5] So was P_{243}, but it is difficult to understand why it should be so, in general.

4.3.3 Homonym Detection

Since our research focused primarily on the synonym problem, and since that is the harder one, we will address the homonym problem only briefly. In short, our procedures did an excellent job of detecting the 3 homonym problems. There were no type 1 errors for any procedure, and only procedures P_{3jk} had type 2 errors when the correct quiddities were used. With the quiddities obtained in the experiment, procedures P_{111} and P_{211} each had an average of 0.2 type 2 errors, and several other procedures had an average of less than 1. It seems logical to conclude that the stricter procedures P_{111} and P_{211} are best suited to the detection of homonym problems.

5 Conclusions

The basic principle underlying our strategy for detecting naming conflicts is that this process must rely on semantic information about the data elements in the database. We believe that the concept of quiddities, as defined in Bhargava et al. [4] and as refined in this paper, can effectively capture the semantic information necessary for the detection of these conflicts. Our experiments, though conducted on a small scale. do provide evidence to support this belief. We found that database users could be trained to declare quiddity information accurately enough that it could be used by inference procedures to detect naming conflicts. Certain of our inference procedures performed reasonably well in detecting these conflicts. However, much more testing needs to be done before any general conclusions can be reached from these results.

There are several ways to obtain higher accuracy and consistency in quiddity specification. One is to refine the definitions of the quiddity categories and leave the burden on the user to develop correct and more accurate quiddities. The second is to shift the burden to the inference procedures, by defining sophisticated procedures that take into account inconsistencies such as differences in the level of detail or specificity. Our approach is a combination of these two, but emphasizes the latter. A third is to develop interactive, automated tools for supporting the quiddity acquisition process. All of these alternatives, and particularly the last one, are issues for further research. The "quiddities approach" for the detection of naming conflicts is fundamentally different from other approaches in its use of formalized semantic information in conjunction with automated inference procedures. It would be interesting to examine if and how this strategy could be applied to other aspects of database integration as well.

References

[1] Batini, C. and M. Lenserini, "A Methodology for Data Schema Integration in the Entity Relationship Model," *IEEE Transactions on Software Engineering,* **SE-10: 6** pp. 650–663, 1984.

[2] Batini, C., M. Lenserini, and S. Navathe, "A Comparative Analysis of Methodologies for Database Schema Integration," *ACM Computing Surveys,* **18: 4**, pp. 323–364, 1986.

[3] Beyer, R.M., "The Problem of Unique Name Violations in Database Integration," Master's thesis, Naval Postgraduate School, Monterey, March 1991.

[4] Bhargava, H.K., S.O. Kimbrough, and R. Krishnan, "Unique Names Violations, a Problem for Model Integration or You Say Tomato, I Say Tomahto," *ORSA Journal on Computing,* forthcoming, 1991.

[5] Bouzeghoub, M., G. Gardarin, and E. Metais, "Database Design Tools: An Expert System Approach," *Proceedings of Very Large Databases,* pp. 82–94, 1985.

[6] Casanova, M., and M. Vidal, "Towards a Sound View Integration Methodology," *Proceedings of the* 2^{nd} *ACM SIGACT / SIGMOD Conference on the Principles of Database Systems,* pp. 36–47, 1983.

[7] Choobineh, J., M.V. Mannino, J.F. Nunamaker, B.R. Konsynski, "An Expert Database Design System Based on Analysis of Forms," *IEEE Transactions on Software Engineering,* **14: 2**, February 1988.

[8] ElMasri, R., J. Larson, and S. Navathe, "Integration Algorithms for Federated Databases and Logical Database Design," Technical Report, Honeywell Corporate Research Center, 1987.

[9] Genesereth and Nils J. Nilsson, *Logical Foundations of Artificial Intelligence,* Morgan Kaufmann Publishers, Inc., Palo Alto, California, 1987.

[10] Hayne, S., and S. Ram, "Multi-User View Integration System (MUVIS): An Expert System for View Integration," *Proceedings of the Sixth International Conference on Data Engineering,* pp. 402–409, February 1990.

[11] Kamel, M.N., and D.K. Hsiao, "Interoperability and Integration Issues in Heterogeneous Database Environments," *Information Systems Research,* forthcoming, 1991.

[12] Larson, J.A., S.B. Navathe, and R. Elmasri, "A Theory of Attribute Equivalence in Databases with Application to Schema Integration," *IEEE Transactions on Software Engineering,* **15: 4**, pp. 449–463, April 1989.

[13] Mannino, M.V., and W. Effelsberg, "A Methodology for Global Schema Design," *Technical Report TR-84-1,* Computer and Information Sciences Department, University of Florida, 1984.

[14] Sterling, L. and E. Shapiro, *The Art of Prolog: Advanced Programming Techniques*, The MIT Press, Cambridge, Massachusetts, 1986.

[15] Wang, Y.R., and S.E. Madnick, "The Inter-Database Instance Identification Problem in Integrating Autonomous Systems," *Proceedings of the Fifth International Conference on Data Engineering,* pp. 46–55, February 1989.

[16] Yao, S.B., V. Waddle and B. Housel, "View Modeling and Integration Using the Functional Data Model," *IEEE Transactions on Software Engineering,* **SE-8: 6**, pp. 544–553, 1982.

[17] Zviran, M., and M.N. Kamel, "A Methodology for Integrating Heterogeneous Databases in a Hospital Environment," Working Paper No. 89-10, Department of Administrative Sciences, Naval Postgraduate School, Monterey, CA, 1989.

Distribution List

Agency	No. of copies
Defense Technical Information Center Cameron Station Alexandria, VA 22314	2
Dudley Knox Library, Code 0142 Naval Postgraduate School Monterey, CA 93943	2
Office of Research Administration Code 08 Naval Postgraduate School Monterey, CA 93943	1
Library, Center for Naval Analyses 4401 Ford Avenue Alexandria, VA 22302-0268	1
Department of Administrative Sciences Library Code AS Naval Postgraduate School Monterey, CA 93943	1
Department of Administrative Sciences Code AS/Bh Naval Postgraduate School Monterey, CA 93943	5